1 Methodology

This paper seeks to contribute to the small literature of interpretable fake news detection, by following the methodology of Horne et al. and Gruppi et al., leveraging features that describe textual properties of fake news, applying non-neural network models and focusing heavily on interpretation of results. This section details the precise methodology employed, discussing the dataset used, the feature engineering process, outlier removal, and models applied.

1.1 Dataset

There are many datasets freely available for fake news detection, but many suffer quality limitations. Oshikawa and Qiang outline 12 requirements for a quality fake news dataset, expanding a 9-point list originally by Rubin et al.: 1. Availability of both truthful and deceptive instances; 2. Digital textual format acessibility; 3. Verifiability of ground truth; 4. Homogeneity in lengths; 5. Homogeneity of writing mattes; 6. Predefined timeframe; 7. The manner of news delivery; 8. Pragmatic concerns; 9. Language and culture; 10. Easy to create from raw data; 11. Fine-grained truthfulness; 12. Various sources or publishers.[1] While no dataset currently exists that meets all 12 criteria, the article-level *FakeNewsNet* (FNN) dataset meets most.[2]

FNN is an article-level dataset that includes the title and body of each article (2), each of which have been profesionally fact-checked and labeled as true or false by Politifact or Gossicop(1, 3). FNN provides both political and celebrity news articles, but this paper chooses to use only the political articles in order to maintain a roughly consistent corpus (4,5,7). These articles are all in English, largely center around American politics, and come from a variety of sources (9, 12). Finally, there is little work needed to obtain the dataset, as FNN provides an API to quickly obtain the body and title of each article (10). The biggest limitation for FNN is that it lacks a fine-grained scale of truth, labeling only as binary

[1]Oshikawa, Rubin
[2]FNN

true/false (11). However, given that it meets the majority of the criteria, and contains 726 observations, it is overall a good fit for this paper.

1.2 Feature Engineering

While FNN includes a host of metadata on each article, this paper utilizes only features engineered from the text and titles of each article. The objective of this paper is to reach new conclusions about the content of fake news, making certain metadata irrelevant or problematic. For instance, the usage of website traffic to gauge veracity may increase accuracy,[3] but measures nothing about the actual content of the article. It should be obvious that websites with high traffic are perfectly capable of producing misinformation. Aditionally, equating established sources with reliable sources can be problematic, failing to hold mainstream media accountable and preventing the growth of new, quality publishers.

The choice of using features capturing exclusively textual properties is motivated by a belief that the problem of fake news cannot be solved with deep learning models automatically policing all the content on the internet. A widespread system like this would have incredible power, and could easily become a force for opression and misinformation with biased data or improper use. Aditionally, people are unlikely to be convinced that something they believe to be true is misinformation just because a browser extension tells them it is. Fake news is a social problem, and will only be fixed with widespread education on how to identify fake news. To achieve this, this paper treats fake news detection as a learning opportunity, focusing solely on identifying the textual properties of text that might suggest an article is misinformation.

This paper uses the FNN dataset to obtain the title and body of articles as well as a true/false label, then leverages a series of natural language processing tools to engineer features describing the text. Each feature is calculated for both the body and the title seperately. Each feature falls under one of four categories: 1) complexity metrics, 2) summary, grammatical, and psychological metrics from the Linguistic Inquiry and Word Count engine,[4] 3) parts-of-speech tagging, and 4) named entity recognition. Each category represents a different method or tool for feature engineering. Each feature engineered is displayed in the tables below.[5]

[3]forget which one does this

[4]

[5]And described in detail in Appendix A

Table 1.1: Complexity Metrics

Variable	Description
mu_sentence	Mean number of sentences
mu_verb_phrase	Mean depth of verb-phrase trees
mu_noun_phrase	Mean depth of noun-phrase trees
sd_sentence	Standard deviation of number of sentences
sd_verb_phrase	Standard deviation of depth of verb-phrase trees
sd_noun_phrase	Standard deviation of depth of noun-phrase trees
iqr_sentence	Interquantile range of number of sentences
iqr_verb_phrase	Interquantile range of depth of verb-phrase trees
iqr_noun_phrase	Interquantile range of depth of verb-phrase trees
num_verb_phrase	Number of verb-phrase trees
swc	Mean sentence word count
wlen	Mean word length
types	Number of unique words
tokens	number of total words
TTR	Type-token ration
FOG	Gunning's Fog Index
SMOG	Simple Measure of Gobbledygook
FK	Flesch-Kincaid Readability Score
CL	Coleman-Liau Index
ARI	Automated Readability Index

The complexity metrics are calculated in three different ways. Several of these metrics are indexes of textual complexity calculated using the "'quanteda"' package in R. Others are variables describing the structure of verb-phrase and noun-phrase trees obtained for each sentence using the Stanford CoreNLP constituency parser. The final complexity metric is a manually computed type-token ratio, where types are all the unique words in a document and tokens are the total words in that document, capturing the diversity of the vocabulary used.

The LIWC features comprise a diverse range of different metrics, many capturing the different psychological components within the text such as cognitive processeses, or core drives and needs.[6]. Some LIWC variables capture simpler properties, such as the frequency

[6]For a full description of each of these variables, consult Appendix A or the LIWC Operator's Manual available at

of informal speech or specific punctuation marks. As LIWC's metrics are heavily dependent on counting words from dictionaries, each metric is normalized by the text of the document, providing a sense of how frequently types of words occur in a standard document size.

All features falling under the third and fourth categories were obtained using the Stanford CoreNLP toolkit. Specifically, the grammatical incidence variables were obtained using the CoreNLP parts-of-speech tagger, which counts the frequency of types of words (for instance, verbs) within a document. As before, these metrics were normalized to account for differing document lengths. Finally, the named entity recognition feature is obtained using the CoreNLP named entity recognition annotator, which simply counts the total number of words that refer to named proper nouns.

1.3 Outlier Removal

The complete data thus consisted of 726 observations with 152 features each, constructed using the body and title of each article obtained by the FNN API, which queries the stored article URLs. However, given that web pages often change structure, move to different addresses, or are removed from the internet entirely, many of the entries have changed since the dataset was initially compiled, and are now unavaiable or in an incorrect format and must be removed from the dataset. To identify outliers, a baseline logistic regression model using all features was fit in order to identify overly influential observations with Cook's Distance four times larger than the mean. This approach cannot perfectly identify all outliers, but suggests that these observations are *potential* outliers. Each of the 575 potential outliers were manually inspected and labeled as either false positives or true outliers. Out of the 575 potential outliers, 422 were true outliers, 132 of which were Politifact articles, bringing the number of observations down to 594.

1.4 Data Analysis and Modeling

After having completed all feature engineering and outlier removal, exploratory data analysis was performed to identify group differences in each predictor between true and false articles. This approach reflects the work Horne et al. and Gruppi et al., allowing for the comparison of results between shared predictors. However, instead of applying an ANOVA test to each predictor, a Mood's Median hypothesis test was performed using the "'RVAdeMemoire"' package[7] as many of the predictors did not meet the normality assumption required by

[7]

ANOVA and other traditional tests. Aditionally, estimates and 95% confidence intervals for each predictor accross both groups were calculated using bootstrapping 1,000 samples.[8]

Given the extensive number of predictors and the multicolinearity between many of them, a binomial Lasso regression—also known as logistic regression with L1 regularization—was applied to avoid overfitting by creating a more parsimonious model. Aditionally, to avoid skewed accuracy results due to class imbalance, the dataset was upsampled to have an equal proportion of negative and positive labels. The upsampling increased the number of negative articles to have a 374-374 split, as opposed to the original 374-220 split. After fitting the Lasso model, the features reduced to zero were removed and a logistic regression model using only the preserved features. This is the model used for final interpretation of results. Aditionally, seperate models are fit for the body and title of articles. An approached focused entirely on predictive accuracy would likely instead create a two-stage ensemble model, but as the overall objective is interpretation, maintaining the two seperate is preferable as it allows for better interpretation of the results.

[8]The results of the Mood's Median tests are displayed in the next section, but the confidence intervals are only shown in Appendix A.